Exercises

A.1 Consider the parallel execution of the three program fragments below.

while ($true$**) { A**: $x := x+2;$ **}**
while ($true$**) { B**: $y := y-1;$ }
while($true$**) { C**: $x := x-1;$ **D**: $y := y+2;$ }

Initially, $x = y = 0$

Synchronize the system in order to maintain

$I0: 0 \leq y$

$I1: x \leq 10$

Provide an argument for the absence of deadlock in the synchronized solution? Optional: can you think of an additional restriction that could cause a deadlock?

Exercises

A.2 Solve the *Producer-Consumer* problem presented in the lecture slides.

What if there are several *Producers* and several *Consumers?*

Exercises

A.3 Given *N* tasks of the form

while (*true***) {** *X(n);* **}**

Where n=0 for the first process, n=1 for the second process, etc. Assume *X(n)* is a non-atomic program section that must be executed under mutual exclusion.

Synchronize this system such that:

a. the sections are executed one after the other, in order:

X(0); X(1); X(2); ....; X(N-1); X(0), X(1), ...

b. *X(i)* is executed at least as often as *X(i+1),* for $0 \leq i < N-1$.

A complete solution must first state appropriate synchronization conditions.

Exercises

A.4 Given a collection of tasks using system procedures *A0* and *A1*. Synchronize the execution of these procedures such that exclusion is provided and that one execution of *A0* and two executions of *A1* alternate:

A0; A1; A1; A0; A1; A1 …

- Is there any danger of deadlock?
- What about the fairness?

Exercises

A.6 Given *N* tasks of the following form

Proc *Philosopher* $(n \mid 0 \le n < N)$ =
|[**while** *(true)* **{**
    NonCriticalSection(n);
    CriticalSection(n)
  }
]|

The critical sections requires to use two resources out of a total of *N* resources; *Philosopher(n)* uses resources number *n* and *n+1* (with addition modulo *N)*. Solve this problem to avoid deadlocks and ensure fairness.

Exercises

A.7 Consider the parallel execution of the three program fragments below.

while(*true)* **{ A0**: $x := x+2;$ **A1**: $y := y-1;$ **A2**: $z := z-1;$ }

while (*true)* **{ B**: $y := y+2;$ **}**

while (*true)* **{ C0**: $z := z+1;$ **C1**: $x := x-2;$ **}**

Initially, $x = y = z = 0$

Synchronize the system in order to maintain

$I0: x+y+z \leq 10$

$I1: y \leq 5$

The direct solution may lead to a deadlock. Give a trace. Can you solve the issue by adding additional restrictions?

Answers to exercises 2INC0: A.1i

Program topology: $x = 2\underline{c}A - \underline{c}C$ and $y = 2\underline{c}D - \underline{c}B$

Sync cond: $I0: \underline{c}B \leq 2\underline{c}D$ and $I1: 2\underline{c}A \leq 10 + \underline{c}C$

Semaphores: s for $I0$, initially $s = 0$

t for $I1$, initially $t = 10$

P1: while(*true* **){** $P^2(t)$; A: $x := x+2$ **}**

P2: while(*true* **){** $P(s)$; B: $y := y-1$ **}**

P3: while(*true* **){**

C: $x := x-1$; $V(t)$

D: $y := y+2$; $V^2(s)$

}

Rudolf Mak, r.h.mak@tue.nl
TU/e , Computer Science, SAN

Answers to exercises 2INC0: A.1ii

Deadlock means that a group of processes are blocked indefinitely.

Since P3 does not contain *P*-operations, it is never blocked. Now, by contradiction, assume that either P1 or P2 is blocked on their *P*-operation on semaphores *t* and *s*, respectively. Since P3 is not blocked and always eventually execute a *V*-operation on *s* and *t*, P1 and P2 cannot be blocked indefinitely on their *P*-operations.

Additional restrictions that may cause deadlock:

1. $x = y$
2. $x \leq y \leq x+1$

Additional restrictions that do not cause a deadlock:

1. $a \times x \leq y$
2. $a \times y \leq x$

Rudolf Mak, r.h.mak@tue.nl
TU/e , Computer Science, SAN

Answers to exercises 2INC0: A.3a

Synchronization conditions:

I_0: $\underline{c}X(0) \leq 1 + \underline{c}X(N\text{-}1)$
I_1: $\underline{c}X(1) \leq \underline{c}X(0)$

.

.

.

$I_{N\text{-}1}$: $\underline{c}X(N\text{-}1) \leq \underline{c}X(N\text{-}2)$

Semaphore s_n **for condition** $\mathbf{I_n}$

Initially: $s_0 = 1$

$s_n = 0, \;\; 1 \leq n \leq N$

$X(0) \rightarrow P(s_0); X(0); V(s_1)$

.

.

.

$X(N\text{-}2) \rightarrow P(s_{N\text{-}2}); X(N\text{-}2); V(s_{N\text{-}1})$
$X(N\text{-}1) \rightarrow P(s_{N\text{-}1}); X(N\text{-}1); V(s_0)$

Rudolf Mak, r.h.mak@tue.nl
TU/e , Computer Science, SAN

Answers to exercises 2INC0: A.3b

Drop synchronization condition I_0 and sem s_0

Synchronization conditions:

I_1: $\underline{c}X(1) \le \underline{c}X(0)$

.

.

.

I_{N-1}: $\underline{c}X(N-1) \le \underline{c}X(N-2)$

Semaphore s_n **for SC** I_n
Initially: $s_n = 0, \; 1 \le n \le N$

$X(0) \rightarrow X(0); V(s_1)$

.

.

.

$X(N-2) \rightarrow P(s_{N-2}); X(N-2); V(s_{N-1})$

$X(N-1) \rightarrow P(s_{N-1}); X(N-1)$

No P

No V

Rudolf Mak, r.h.mak@tue.nl
TU/e , Computer Science, SAN

Answers to exercises 2INC0: A.4i

- **Synchronization conditions:**
- *$\underline{c}A1 \leq 2\underline{c}A0$* **sem: *s*, initially** *$s = 0$*
- *$2\underline{c}A0 \leq 2+\underline{c}A1$* **sem: *t*, initially** *$t = 2$*

- *$A0 \rightarrow P^2(t); A0; V^2(s)$*
- *$A1 \rightarrow P(s); A1; V(t)$*

- **Careful, we must also ensure mutual exclusion between calls to *A0* and *A1*:**

Answers to exercises 2INC0: A.4ii

- **sem**: *m0*, **initially** *m0 = 1*
- *m1*, **initially** *m1 = 1*

- $A0 \rightarrow P^2(t); P(m0); A0; V(m0); V^2(s)$ ← **danger of Deadlock** (see greedy consumers in lecture slides)
- $A1 \rightarrow P(s); P(m1); A1; V(m1); V(t)$

- ***Solution***
- $A0 \rightarrow P(m0); P^2(t); A0; V(m0); V^2(s)$
- $A1 \rightarrow P(s); P(m1); A1; V(m1); V(t)$

Answers to exercises 2INC0: A.6i

Model each resource by a binary semaphore f_n: $0 \le n < N$, initially set to 1

When all philosophers try to acquire their forks in the same order $P(f_n); P(f_{n+1})$, then a deadlock occurs when they all succeed in performing their first P-operation. This can be resolved by

1. A single philosopher, say number $N-1$ that executes its P-operations in the reverse order $P(f_0); P(f_{N-1})$
2. Philosopher n performs $P(f_n); P(f_{n+1})$ when n is even and $P(f_{n+1}); P(f_n)$ when n is odd

These solutions are fair when strong semaphores are used.

Otherwise starvation can occur.

Answers to exercises 2INC0: A.6ii

We prove absence of deadlock for solution 1. Let

DL ≡ *"All processes are blocked on a P-operation"*

$\quad DL$ *and* *phil N-1 blocked* $P(f_0)$
$\Rightarrow$ DL *and* *phil N-1 blocked* $P(f_0)$ *and* *phil 0 blocked* $P(f_1)$
$\quad$ . . .
$\Rightarrow$ DL *and* *phil N-1 blocked* $P(f_0)$ *and* *phil N-2 blocked* $P(f_{N-1})$
$\Rightarrow$ *phil N-2 blocked* $P(f_{N-1})$ *and* $\underline{c}P(f_{N-1}) = \underline{c}V(f_{N-1})$
$\Rightarrow$ *phil N-2 blocked* $P(f_{N-1})$ *and* $f_{N-1} = 1$
$\Rightarrow$ *false*

Similarly, we derive a contradiction when *phil N-1* is blocked on $P(f_{N-1})$

Rudolf Mak, r.h.mak@tue.nl
TU/e , Computer Science, SAN

Answers to exercises 2INC0: A.7i

Program topology: $x = 2\underline{c}A0 - 2\underline{c}C1$ and $y = 2\underline{c}B - \underline{c}A1$ and $z = \underline{c}C0 - \underline{c}A2$
Synchronization conditions: *SC0* for *I0* and *SC1* for *I1.*
SC0: $2\underline{c}A0 + 2\underline{c}B + \underline{c}C0 \le 10 + \underline{c}A1 + \underline{c}A2 + 2\underline{c}C1$
SC1: $2\underline{c}B \le 5 + \underline{c}A1$
Semaphores: *s* for *I0*, initially $s = 10$
t for *I1*, initially $t = 5$

P1: **while(** *true* **){** $P^2(s)$; **A0**: $x := x+2$; **A1:** $y := y-1$; $V(s)$; $V(t)$; **A2:** $z := z-1$; $V(s)$; **}**
P2: **while(** *true* **){** $P^2(t)$; $P^2(s)$; **B:** $y := y+2$; **}**
P3: **while(** *true* **){** $P(s)$;**C0:** $z := z+1$; **C1:** $x := x-2$; $V^2(s)$; **}**

Deadlock scenario: $\mathbf{(A0A1A2)^5}$ $\{x=10 \wedge y=-5 \wedge z=-5\}$ $\mathbf{B^5}$ $\{x+y+z=10 \wedge y=5\}$

Rudolf Mak, r.h.mak@tue.nl
TU/e , Computer Science, SAN

Answers to exercises 2INC0: A.7ii

Program topology: $x + 2z = 2((\underline{c}A0 - \underline{c}A2) + (\underline{c}C0 - \underline{c}C1))$
Topology invariants: $0 \leq \underline{c}A0 - \underline{c}A2 \leq 1$ and $0 \leq \underline{c}C0 - \underline{c}C1 \leq 1$
Hence we have system invariant *I2:* $0 \leq x + 2z \leq 4$

Let *I3:* $y \leq z + 6$, then *I2* ***and*** *I3 => I0.* Hence realize *I1* ***and*** *I3.*

Semaphore: *u* for *I3*, initially $u = 6$

P1: **while(** *true* **){ A0**: *x := x+2;* **A1:** *y := y-1; V(t); V(u);P(u);* **A2:** *z := z-1;* **}**
P2: **while(** *true* **){** *P^2(u); P^2(t);* **B:** *y:= y+2;* **}**
P3: **while(** *true* **){ C0:** *z:= z+1; V(u);* **C1:** *x := x-2;* **}**

Rudolf Mak, r.h.mak@tue.nl
TU/e , Computer Science, SAN